EXHIBIT A

EXHIBIT A

Patent
Attorney Docket No. 432383600013

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	Levergood et al.
Serial No.:	09/005479
Filing Date:	January 12, 1998
For:	Internet Server Access Control and Monitoring Systems
Art Unit:	2145
Examiner:	Patrice L. Winder

Declaration of Prior Invention to Overcome Cited Reference(s) Under 37 C.F.R. § 1.131

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited today with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450
On October 19, 2006.

By: Suzanne Koston ***Suzanne Koston***

Sir:

This Declaration is submitted to establish invention of the claimed subject matter of this application prior to April, 1995 (the alleged "effective date") of the Kahan reference. Claims 3, 35, and 112 are independent claims.

We, Thomas Mark Levergood, Lawrence C. Stewart, Stephen Jeffrey Morris, Andrew C. Payne, and George Winfield Treese, inventors of the subject matter described and claimed in this application, titled "Internet Server Access Control and Monitoring Systems," declare as follows:

1. Prior to the effective date, Messrs. Stewart, Payne and Treese engaged in discussions regarding the development of technology for an Internet-based electronic commerce system for Open Market Inc. ("OMI"), the original assignee of this application. Messrs. Morris and Levergood joined OMI in January 1995. Schedule 1 was generated prior to the effective date and shows conception of claim 3 before the effective date. (The redacted dates in Schedule 1 are prior to the effective date.) In general, Schedule 1 is an e-mail message from one of the inventors (Lawrence C. Stewart) that describes an Internet web-based service for customers and advertisers. As shown on page 2 of Schedule 1, the web-based service can allow customers to use their client browsers to access remote computer servers for purchasing items:

> We've taken the tack of separating authentication from payment from delivery. Typically, a buyer with a browser clicks on a hypertext link we call a Payment URL, which contains the URL of a script on the authentication and payment server, and which also contains a digitally signed representation of the price, and what is being offered for sale. The client browser presents this payment URL to the authentication server, which uses some collection of authentication methods (starting with Basic and working up to smart cards) to establish the identity of the buyer.
> At the moment, the authentication and payment functions are colocated, so the authentication/payment server then enters into its transaction database the fact that this specific customer has purchased <whatever>.
> [(See Schedule 1, page 2)]

As shown by this passage, Schedule 1 describes a "*method of processing service requests from a client to a server system through a network*" (as recited in claim 3). Other passages from Schedule 1 also describe this, such as on page 5 of Schedule 1:

> Server to Client:
> ==========

HTTP/1.0 401 Unauthorized
Content-type: text/html
WWW-Authenticate: Basic+ encoded_string
[...]
Client back to Server:
==========
Authorization: Basic+ encoded_string
==========
Where encoded_string is the RFC 1421 encoding of
username:response:stuff_for_convenience_of_server

This passage from Schedule 1 describes communications between a client and server (e.g., "Server to Client" and "Client back to Server") over the WWW (world wide web) using HTTP (Hypertext Transfer Protocol).

Schedule 1 describes "*forwarding a service request from the client to the server system,*" such as on page 2 of Schedule 1 (emphasis added):

> We've taken the tack of separating authentication from payment from delivery. Typically, a buyer with a browser clicks on a hypertext link we call a Payment URL, which contains the URL of a script on the authentication and payment server, and which also contains a digitally signed representation of the price, and what is being offered for sale. **The client browser presents this payment URL to the authentication server, which uses some collection of authentication methods (starting with Basic and working up to smart cards) to establish the identity of the buyer.** At the moment, the authentication and payment functions are colocated, so the authentication/payment server then enters into its transaction database the fact that this specific customer has purchased <whatever>.

This passage from Schedule 1 describes a client browser providing a request to purchase an item for sale and provides a payment URL to a server for authenticating the user so that the purchase transaction can proceed. As another example, page 5 of Schedule 1 contains the following passage:

> Client back to Server:
> ==========

Authorization: Basic+ encoded_string
==========
Where encoded_string is the RFC 1421 encoding of
username:response:stuff_for_convenience_of_server

This passage from Schedule 1 describes a request from a client to server (e.g., "Client back to Server") over the WWW (world wide web) using HTTP (Hypertext Transfer Protocol).

Schedule 1 describes "*communications between the client and server system are according to hypertext transfer protocol,*" such as on page 5 of Schedule 1 (emphasis added):

Server to Client:
==========
HTTP/1.0 401 Unauthorized
Content-type: text/html
WWW-Authenticate: Basic+ encoded_string

Schedule 1 describes "*returning a session identifier from the server system to the client,*" such as on page 5 of Schedule 1 (emphasis added):

Server to Client:
==========
HTTP/1.0 401 Unauthorized
Content-type: text/html
WWW-Authenticate: Basic+ **encoded_string**

HTML text to be displayed when the browser can't handle it, or when the user clicks on "do not retry authentication"
==========
Where **"encoded_string"** is the RFC 1421 encoding of
challenge:**stuff_for_convenience_of_server**
[...]
In this system, the "stuff_for_convenience_of_server" is simply a place to stash the "state" of the transaction, so the server doesn't necessarily need a database. We use this for a digital signature of the expected response.

This passage from Schedule 1 describes the server sending to the client (i.e., "Server to Client") an "encoded string" that has information to identify a session, that is information which "stash[es] the 'state' of the session or transaction."

Schedule 1 describes "*the client storing the session identifier for use in subsequent distinct requests to the server system,*" such as on page 5 of Schedule 1 (emphasis added):

> Server to Client:
> ==========
> HTTP/1.0 401 Unauthorized
> Content-type: text/html
> WWW-Authenticate: Basic+ **encoded_string**
>
> HTML text to be displayed when the browser can't handle it, or when the user clicks on "do not retry authentication"
> ==========
> Where "**encoded_string**" is the RFC 1421 encoding of
> challenge:**stuff_for_convenience_of_server**
>
> Client back to Server:
> ==========
> Authorization: Basic+ encoded_string
> ==========
> Where **encoded_string** is the RFC 1421 encoding of
> username:response:**stuff_for_convenience_of_server**
>
> **In this system, the "stuff_for_convenience_of_server" is simply a place to stash the "state" of the transaction, so the server doesn't necessarily need a database.** We use this for a digital signature of the expected response.

This passage from Schedule 1 describes the server providing to the client transaction state information (i.e., "stuff_for_convenience_of_server"). The client receives this information from the server (i.e., "Server to Client") and stores it so that when the client needs to provide a response to the server (i.e., "Client back to Server"), the client can locate the "stuff_for_convenience_of_server" and provide this transaction state information back to the

server along with the response. As mentioned in this passage, the advantage of this approach is that the server is not required to use a database to store the state of the transaction.

Schedule 1 describes "*appending the stored session identifier to each of the subsequent distinct requests from the client to the server system,*" such as on page 5 of Schedule 1 (emphasis added):

> Server to Client:
> ==========
> HTTP/1.0 401 Unauthorized
> Content-type: text/html
> WWW-Authenticate: Basic+ encoded_string
>
> HTML text to be displayed when the browser can't handle it, or when the user clicks on "do not retry authentication"
> ==========
> Where "encoded_string" is the RFC 1421 encoding of
> challenge:stuff_for_convenience_of_server
>
> Client back to Server:
> ==========
> **Authorization: Basic+ encoded_string**
> ==========
> Where encoded_string is the RFC 1421 encoding of
> **username:response:stuff_for_convenience_of_server**
>
> In this system, the "stuff_for_convenience_of_server" is simply a place to stash the "state" of the transaction, so the server doesn't necessarily need a database. We use this for a digital signature of the expected response.

This passage from Schedule 1 describes that for a response from a "Client back to Server", the client provides the "encoded_string" to the server. The "encoded string" contains not only the username and the response, but coupled thereto is the "stuff_for_convenience_of_server" which contains the state information. If any subsequent response does not contain such information, then the subsequent transaction request would fail since the request would not contain the proper authorization information.

Accordingly as shown by the above, Schedule 1 shows conception of claim 3 before the effective date and thus the Kahan reference should be removed as a reference.

2. Schedule 1 shows conception of claim 35 before the effective date. More specifically, Schedule 1 describes "*an information system on a network,*" such as on page 5 of Schedule 1:

> Server to Client:
> ==========
> HTTP/1.0 401 Unauthorized
> Content-type: text/html
> WWW-Authenticate: Basic+ encoded_string
> [...]
> Client back to Server:
> ==========
> Authorization: Basic+ encoded_string
> ==========
> Where encoded_string is the RFC 1421 encoding of
> username:response:stuff_for_convenience_of_server

This passage from Schedule 1 describes communications between a client and server (e.g., "Server to Client" and "Client back to Server") over the WWW (world wide web) using HTTP (Hypertext Transfer Protocol).

Schedule 1 describes "*means for receiving service requests from a client and for determining whether a service request includes a session identifier,*" such as on page 5 of Schedule 1:

> Client back to Server:
> ==========
> Authorization: Basic+ encoded_string
> ==========
> Where encoded_string is the RFC 1421 encoding of
> username:response:stuff_for_convenience_of_server

> In this system, the "stuff_for_convenience_of_server" is simply a place to stash the "state" of the transaction, so the server doesn't necessarily need a database. We use this for a digital signature of the expected response.

This passage from Schedule 1 describes the client providing a response (i.e., "Client back to Server") to the server that includes the "stuff_for_convenience_of_server" (i.e., the "state" of the transaction). As mentioned in this passage, the advantage of this approach is that the server is not required to use a database to store the state of the transaction. Since the server does not have to store the transaction state information, the server uses the stuff_for_convenience_of_server provided by the client in order to obtain the transaction state information. Moreover the server checks to see if the session identifier has been forged or tampered with by using "this for a digital signature of the expected response."

Schedule 1 describes "*wherein communications to and from the client are according to hypertext transfer protocol*," such as on page 5 of Schedule 1 (emphasis added):

> Server to Client:
> ==========
> **HTTP**/1.0 401 Unauthorized
> Content-type: text/html
> WWW-Authenticate: Basic+ encoded_string

Schedule 1 describes "*means for providing the session identifier in response to an initial service request from the client in a session of requests*," such as on page 5 of Schedule 1 (emphasis added):

> Server to Client:
> ==========
> HTTP/1.0 401 Unauthorized
> Content-type: text/html
> WWW-Authenticate: Basic+ **encoded_string**
>
> HTML text to be displayed when the browser can't handle it, or when the user clicks on "do not retry authentication"

==========

Where **"encoded_string"** is the RFC 1421 encoding of
challenge:**stuff_for_convenience_of_server**
[...]
In this system, the "stuff_for_convenience_of_server" is simply a place to stash the "state" of the transaction, so the server doesn't necessarily need a database. We use this for a digital signature of the expected response.

This passage from Schedule 1 describes the server sending to the client (i.e., "Server to Client") an "encoded string" that has information to identify a session, that is information which "stash[es] the 'state' of the session or transaction."

Schedule 1 describes "*means for storing, at the client, the session identifier for use in each communication to the server system,*" such as on page 5 of Schedule 1 (emphasis added):

Server to Client:
==========
HTTP/1.0 401 Unauthorized
Content-type: text/html
WWW-Authenticate: Basic+ **encoded_string**

HTML text to be displayed when the browser can't handle it, or when the user clicks on "do not retry authentication"
==========
Where "**encoded_string**" is the RFC 1421 encoding of
challenge:**stuff_for_convenience_of_server**

Client back to Server:
==========
Authorization: Basic+ encoded_string
==========
Where **encoded_string** is the RFC 1421 encoding of
username:response:**stuff_for_convenience_of_server**

In this system, the "stuff_for_convenience_of_server" is simply a place to stash the "state" of the transaction, so the server doesn't necessarily need a database. We use this for a digital signature of the expected response.

This passage from Schedule 1 describes the server providing to the client transaction state information (i.e., "stuff_for_convenience_of_server"). The client receives this information from the server (i.e., "Server to Client") and stores it so that when the client needs to provide a response to the server (i.e., "Client back to Server"), the client can locate the "stuff_for_convenience_of_server" and provide this transaction state information back to the server along with the response. As mentioned in this passage, the advantage of this approach is that the server is not required to use a database to store the state of the transaction.

Schedule 1 describes "*means for appending the stored session identifier to each of subsequent communications from the client to the server system,*" such as on page 5 of Schedule 1 (emphasis added):

> Server to Client:
> ==========
> HTTP/1.0 401 Unauthorized
> Content-type: text/html
> WWW-Authenticate: Basic+ encoded_string
>
> HTML text to be displayed when the browser can't handle it, or when the user clicks on "do not retry authentication"
> ==========
> Where "encoded_string" is the RFC 1421 encoding of
> challenge:stuff_for_convenience_of_server
>
> Client back to Server:
> ==========
> **Authorization: Basic+ encoded_string**
> ==========
> Where encoded_string is the RFC 1421 encoding of
> **username:response:stuff_for_convenience_of_server**
>
> In this system, the "stuff_for_convenience_of_server" is simply a place to stash the "state" of the transaction, so the server doesn't necessarily need a database. We use this for a digital signature of the expected response.

This passage from Schedule 1 describes that for a response from a "Client back to Server", the client provides the "encoded_string" to the server. The "encoded string" contains not only the username and the response, but coupled thereto is the "stuff_for_convenience_of_server" which contains the state information. If any subsequent response does not contain such information, then the subsequent transaction request would fail since the request would not contain the proper authorization information.

Schedule 1 describes "*means for servicing the subsequent service requests*," such as on page 2 of Schedule 1:

> Section 2 - relevant facts about OMI's systems
> We've taken the tack of separating authentication from payment from delivery. Typically, a buyer with a browser clicks on a hypertext link we call a Payment URL, which contains the URL of a script on the authentication and payment server, and which also contains a digitally signed representation of the price, and what is being offered for sale.
>
> The client browser presents this payment URL to the authentication server, which uses some collection of authentication methods (starting with Basic and working up to smart cards) to establish the identity of the buyer.
>
> At the moment, the authentication and payment functions are colocated, so the authentication/payment server then enters into its transaction database the fact that this specific customer has purchased <whatever>. The payment server then issues an HTTP redirect to the client, passing an "Access URL".
>
> The Access URL is the URL for the document purchased, together with digitally signed information about the expiration date of the access, and (at the moment) the IP address of the purchaser.
>
> The Access URL is presented by the client to the server which contains the document purchased. This "content server" validates the access URL, and if it is valid, returns the document to the client.

This passage describes a series of client requests (e.g., a customer requests for purchasing "<whatever>") which the server services.

Accordingly as shown by the above, Schedule 1 shows conception of claim 35 before the effective date and thus the Kahan reference should be removed as a reference.

3. Schedule 1 shows conception of claim 112 before the effective date. More specifically, Schedule 1 describes "*a method of processing, in a server system, service requests from a client to the server system through a network,*" such as on page 5 of Schedule 1:

> Server to Client:
> ==========
> HTTP/1.0 401 Unauthorized
> Content-type: text/html
> WWW-Authenticate: Basic+ encoded_string
> [...]
> Client back to Server:
> ==========
> Authorization: Basic+ encoded_string
> ==========
> Where encoded_string is the RFC 1421 encoding of
> username:response:stuff_for_convenience_of_server

This passage from Schedule 1 describes communications between a client and server (e.g., "Server to Client" and "Client back to Server") over the WWW (world wide web) using HTTP (Hypertext Transfer Protocol).

Schedule 1 describes "*receiving, from the client, a service request to which a session identifier stored at the client has been appended by the client,*" such as on page 5 of Schedule 1 (emphasis added):

> Server to Client:
> ==========
> HTTP/1.0 401 Unauthorized
> Content-type: text/html
> WWW-Authenticate: Basic+ **encoded_string**
>
> HTML text to be displayed when the browser can't handle it, or when the
> user clicks on "do not retry authentication"

==========

Where "**encoded_string**" is the RFC 1421 encoding of
challenge:**stuff_for_convenience_of_server**

Client back to Server:
==========
Authorization: Basic+ encoded_string
==========
Where **encoded_string** is the RFC 1421 encoding of
username:response:**stuff_for_convenience_of_server**

In this system, the "stuff_for_convenience_of_server" is simply a place to stash the "state" of the transaction, so the server doesn't necessarily need a database. We use this for a digital signature of the expected response.

This passage from Schedule 1 describes the server providing to the client transaction state information (i.e., "stuff_for_convenience_of_server"). The client receives this information from the server (i.e., "Server to Client") and stores it so that when the client needs to provide a response to the server (i.e., "Client back to Server"), the client can locate the "stuff_for_convenience_of_server" and provide this transaction state information back to the server along with the response. The "encoded string" contains not only the username and the response, but coupled thereto is the "stuff_for_convenience_of_server" which contains the state information. If any subsequent response does not contain such information, then the subsequent transaction request would fail since the request would not contain the proper authorization information.

Schedule 1 describes "*wherein communications between the client and server system are according to hypertext transfer protocol,*" such as on page 5 of Schedule 1 (emphasis added):

Server to Client:
==========
HTTP/1.0 401 Unauthorized
Content-type: text/html
WWW-Authenticate: Basic+ encoded_string

Schedule 1 describes "*validating the session identifier appended to the service request,*" such as on page 2 of Schedule 1:

> The Access URL is presented by the client to the server which contains the document purchased. This "content server" validates the access URL, and if it is valid, returns the document to the client.

As shown by this passage from Schedule 1, the access information is validated by the content server.

Schedule 1 describes "*servicing the service request if the appended session identifier is valid,*" such as on page 5 of Schedule 1:

> Client back to Server:
> ==========
> Authorization: Basic+ encoded_string
> ==========
> Where encoded_string is the RFC 1421 encoding of
> username:response:stuff_for_convenience_of_server
>
> In this system, the "stuff_for_convenience_of_server" is simply a place to stash the "state" of the transaction, so the server doesn't necessarily need a database. We use this for a digital signature of the expected response.

This passage from Schedule 1 describes the client providing a response (i.e., "Client back to Server") to the server that includes the "stuff_for_convenience_of_server" (i.e., the "state" of the transaction). As indicated in this passage, the server checks to see if the session identifier has been forged or tampered with by using "this for a digital signature of the expected response." If the digital signature is validated, then the service request (e.g., a document) is serviced, such as shown on page 2 of Schedule 1:

> The Access URL is presented by the client to the server which contains the document purchased. This "content server" validates the access URL, and if it is valid, returns the document to the client.

Accordingly as shown by the above, Schedule 1 shows conception of claim 112 before the effective date and thus the Kahan reference should be removed as a reference.

4. Schedule 2 contains an e-mail message from an inventor (Thomas M. Levergood) discussing review of the "Session ID Patent" (see RE: line of the message) which evidences due diligence from prior to the effective date to the filing of the application.

5. We hereby declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

By: _____ Date: 10/13/06
Andrew C. Payne

By: _____ Date: _____
Lawrence C. Stewart

SEE ATTACHED

By: _____ Date: _____
George Winfield Treese

By: _____ Date: _____
Thomas Mark Levergood

By: _____ Date: _____
Stephen Jeffrey Morris

Accordingly as shown by the above, Schedule 1 shows conception of claim 112 before the effective date and thus the Kahan reference should be removed as a reference.

4. Schedule 2 contains an e-mail message from an inventor (Thomas M. Levergood) discussing review of the "Session ID Patent" (see RE: line of the message) which evidences due diligence from prior to the effective date to the filing of the application.

5. We hereby declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

By: _____ Date: SEE ATTACHED
Andrew C. Payne

By: _____ Date: Oct 13, 2006
Lawrence C. Stewart

By: _____ Date: _____
George Winfield Treese

SEE ATTACHED

By: _____ Date: _____
Thomas Mark Levergood

By: _____ Date: _____
Stephen Jeffrey Morris

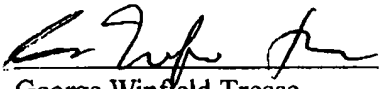
Accordingly as shown by the above, Schedule 1 shows conception of claim 112 before the effective date and thus the Kahan reference should be removed as a reference.

4. Schedule 2 contains an e-mail message from an inventor (Thomas M. Levergood) discussing review of the "Session ID Patent" (see RE: line of the message) which evidences due diligence from prior to the effective date to the filing of the application.

5. We hereby declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

By: _____ Date: _____
Andrew C. Payne
SEE ATTACHED

By: _____ Date: _____
Lawrence C. Stewart

By:  Date: 10/13/2006
George Winfield Treese

By: _____ Date: _____
Thomas Mark Levergood
SEE ATTACHED

By: _____ Date: _____
Stephen Jeffrey Morris

Accordingly as shown by the above, Schedule 1 shows conception of claim 112 before the effective date and thus the Kahan reference should be removed as a reference.

4. Schedule 2 contains an e-mail message from an inventor (Thomas M. Levergood) discussing review of the "Session ID Patent" (see RE: line of the message) which evidences due diligence from prior to the effective date to the filing of the application.

5. We hereby declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

By: _____ Date: _____
Andrew C. Payne
SEE ATTACHED

By: _____ Date: _____
Lawrence C. Stewart

By: _____ Date: _____
George Winfield Treese

By: _____ Date: Oct 12, 2006
Thomas Mark Levergood

SEE ATTACHED

By: _____ Date: _____
Stephen Jeffrey Morris

Accordingly as shown by the above, Schedule 1 shows conception of claim 112 before the effective date and thus the Kahan reference should be removed as a reference.

4. Schedule 2 contains an e-mail message from an inventor (Thomas M. Levergood) discussing review of the "Session ID Patent" (see RE: line of the message) which evidences due diligence from prior to the effective date to the filing of the application.

5. We hereby declare that all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

By: _____ Date: _____
Andrew C. Payne

SEE ATTACHED

By: _____ Date: _____
Lawrence C. Stewart

By: _____ Date: _____
George Winfield Treese

By: _____ Date: _____
Thomas Mark Levergood

By: _____ Date: 10/11/06
Stephen Jeffrey Morris

SCHEDULE 1

Received: from arctic.openmarket.com (arctic.openmarket.com [199.170.183.8]) by relay.openmarket.com [redacted] with ESMTP id UAA21939; [redacted] 20:53:30 -0500
Received: from OpenMarket.com (LOCALHOST [127.0.0.1]) by arctic.openmarket.com [redacted] with ESMTP id UAA04286; Tue, [redacted] -0500
Message-Id: [redacted].UAA04286@arctic.openmarket.com>
To: jeff@spyglass.com, paul@spyglass.com
cc: stewart@OpenMarket.com, timk@spyglass.com, payne@OpenMarket.com,
        treese@OpenMarket.com
Subject: OnAccount (TR-104)
X-Mailer: [redacted]
Date: [redacted]
From: "Lawrence C. Stewart" <stewart@OpenMarket.com>

Agenda:

1. What I thought we were talking about.
2. Relevant facts about OMI's systems
3. Thoughts about TR-104
4. A counterproposal
5. A fancier, but far more wonderful counterproposal

Section 1. - What I thought we were talking about

When Tim was here in Cambridge, he described a slightly fancier authentication
protocol than Basic Auth, in which the server sent a nonce, and the client
sent back the nonce encrypted by the user's password.

The objective was merely (only) to prevent the password running over the net.

I am wildly in favor of this.

Here's how basic authentication works at the server end. Whenever the
server is unhappy about the name and password it has received, it returns

==========
HTTP/1.0 401 Unauthorized
Content-type: text/html
WWW-Authenticate: Basic realm="Whatever"

HTML text to be displayed when the browser can't handle it, or when the
user clicks on "do not retry authentication"
==========

The minimalist change is simply to return

==========
HTTP/1.0 401 Unauthorized
Content-type: text/html

WWW-Authenticate: Basic+ realm="NONCE"

HTML text to be displayed when the browser can't handle it, or when the
user clicks on "do not retry authentication"
==========

Then the browser will retry with the password set to the encrypted nonce.

This mechanism is all we (Open Market) really needs to get more secure
commerce going. (more secure than Basic Authentication).

==========
Section 2 - relevant facts about OMI's systems

We've taken the tack of separating authentication from payment from delivery.
Typically, a buyer with a browser clicks on a hypertext link we call
a Payment URL, which contains the URL of a script on the authentication and
payment server, and which also contains a digitally signed representation of
the price, and what is being offered for sale.

The client browser presents this payment URL to the authentication server,
which uses some collection of authentication methods (starting with
Basic and working up to smart cards) to establish the identity of the buyer.

At the moment, the authentication and payment functions are colocated, so the
authentication/payment server then enters into its transaction database the
fact that this specific customer has purchased <whatever>. The payment
server then issues an HTTP redirect to the client, passing an "Access URL".

The Access URL is the URL for the document purchased, together with
digitally signed information about the expiration date of the access, and
(at the moment) the IP address of the purchaser.

The Access URL is presented by the client to the server which contains the
document purchased. This "content server" validates the access URL, and if
it is valid, returns the document to the client.

2a) Good things about this scheme.

It allows access to one OR a collection of documents given a single
transaction.
It allows subscription (time limited access) to one or a collection of
documents.

2b) Less than perfect things about this scheme.

It correctly handles payment for goods, but after that, access to the document
is mediated by source IP address and possession of the access URL.

2c) Fixes for problems in 2b)

A very small variation allows authentication of the final TCP connection.
I'll come back to this in section 4.

==========

Section 3 - Notes about TR104

1) I think it unneccessarily confuses authentication and payment. Mostly
we want authentication. Occasionally we want payment as well.

2) The semantics about what you get in trade for payment are not wonderful.
It seems that basically the buyer gets the right to attempt to retrieve
a document once, in trade for the purchase.

A) If the server charges the user before sending the document, what is the status of the transaction if the connection breaks before the
document is transmitted?

B) If the server charges the user after the document is successfully
transmitted, then the hacker modifies the browser to drop the connection after
retrieving almost all the document.

C) What happens if the user clicks Reload? What happens if the printer is
out of paper?

[Instead, the OMI system design atomically trades payment for access to
the document. The user can make multiple retrievals not to a single document
but possibly to a whole area, until access expires. If the user loses the
Access URL, a fresh copy of it can be gotten from the payment system's on-line

statement.]

3) What is the purpose for including the price in the encrypted response?
Surely the server doesn't care about this, because the server already knows
the price. I suppose it is a way of ensuring that the price was not
altered in transit, then altered back (man in the middle attack), but
since the URLs are not protected by signatures, why bother protecting
the price?

4) What is the point of the "USD:45:-1" format for money? Is this part
of some ISO standard? Why not "USD:4.50" ?

==========

Section 4 - Counterproposal

What would be easiest for us, and essentially as effective for demonstration
purposes, would be:

1) Basic+ Authentication, as in section 1.

2) OMI authenticaton and payment system used to handle payment at the
contents level, rather than at the HTTP level.
3) OMI "Remote payment" module to
   a) Allow merchant to write payment URLs
   b) Allow merchant to validate access URLs
   This could run on pretty much any CGI capable server
4) Possibly the enahncement referred to above for authenticating the
   final TCP connection.

OMI already has support for authentication using SNK Digital Pathways
cards. The server sends a challenge (digit string) and the client
responds with a digit string representing the challenge encyrpted by
the secret key shared between client and server.

In order to solve the problem of storing the challenge somewhere, the
digital signature of the expected response is also passed along with the
challenge. We do this by passing the signature of the expected response
among the other assorted name value pairs in the URL, but one could
also pass this in the authentication protocol, which would make it more
generic.

This scheme looks like this:

Server to Client:
==========
HTTP/1.0 401 Unauthorized
Content-type: text/html
WWW-Authenticate: Basic+ encoded_string

HTML text to be displayed when the browser can't handle it, or when the
user clicks on "do not retry authentication"
==========
Where "encoded_string" is the RFC 1421 encoding of
challenge:stuff_for_convenience_of_server

Client back to Server:
==========
Authorization: Basic+ encoded_string
==========
Where encoded_string is the RFC 1421 encoding of
username:response:stuff_for_convenience_of_server

In this system, the "stuff_for_convenience_of_server" is simply a place
to stash the "state" of the transaction, so the server doesn't necessarily
need a database. We use this for a digital signature of the expected
response.

A (slight) modification of this protocol changes it as follows:

Server to Client:
==========
HTTP/1.0 401 Unauthorized
Content-type: text/html
Location: $url
WWW-Authenticate: Basic+ encoded_string

HTML text to be displayed when the browser can't handle it, or when the
user clicks on "do not retry authentication"
==========

Notice that this is a combined Redirect and request for authentication.
In this case, the client should send the next request, along with the
authorization string, to the referenced server.

This modification is interesting, because the final step of the authentication

is done on the actual connection that will deliver the requested content,
correcting the problem in Section 2 item 2b.

==========

Section 5 - Fancier counterproposal

We (OMI) also have implemented a Mosaic sidecar application launched by the
MIME mechanism which we call NetPIN. It essentially does all this, by
sending the request-for-authentication information to NetPIN, which does
the client side, then NetPIN passes information back to the server by
causing Mosaic to jump to a NetPIN provided URL which includes some
digital signature stuff in the URL.

This is a pretty good system for doing payment authorizations, beyond doing
straight authentication, because it passes around signatures of the
complete payment information package, including merchant, price, description
of goods, and so forth. This is end-to-end stuff and consequently difficult
to attack by man-in-the-middle methods. The final authorization from
client to server is small, just the "YES" and nonce encrypted by the shared
key.

I would be delighted if we could consider adding this kind of capability
to Spyglass Mosaic. Here's how it would work.

1. Client to server
   <some request>
2. Server to client
   HTML page, with an additional header field:
   WWW-Authenticate: Basic++ <encoded_string>

where encoded_string is the RFC1421 encoding of
nonce:xxx:stuff_for_convenience_of_server

where xxx is the MD5 hash
of the HTML content concatenated with the user's secret key. The secretkey
is not transmitted, but is known to both ends. This is essentially a
digital signature in a symmetric key environment.

[The client needs to validate this signature, which basically assures that

the contents of the screen have not been altered in transit, and also
authenticates
the server, which alone knows the shared secret key]

[The contents of the screen, in the payment context, inform the user of
the charges that would apply to the transaction. The content also includes
a hypertext link that if clicked, will authorize payment, call this link
the "confirm URL"]

3. Client to server
   Request "confirm URL", together with a header field
   Authorization: Basic++ <encoded_string>

where encoded_string is the RFC1421 encoding of
   username:response:stuff_for_convenience_of_server

where response is the MD5 hash of the
nonce concatenated with the confirm URL and the user's secret key

Notes:
A) Startup.

Before the server knows who is the user, what key should it use to send the
"xxx" stuff?

Answer: The xxx stuff should be the empty string, which the client ignores.

B) Redirect

The server-to-client stuff could include a Location:, in which case the
browser could go ahead and automatically generate the Basic++ request
to the redirect location.

C) What does Spyglass have to do to implement this system?

1) Handle the WWW-Authenticate: Basic++ header message:
   a) check the signature, if present
   b) store the stuff_for_convenience_of_server, if present
   c) store the nonce, if present
2) Handle the Authorize: Basic++ header line, if required:
   a) generate the response field, using nonce and URL
   b) pass along the stuff_for_convenience_of_server

D) What does OMI have to do to implement this system?
1) Write the WWW-Authenticate: Basic++ header
   (This is code we already have, I think)
2) Modify the server to validate the Authorize: Basic++ header
   (Also code we nearly have)

What do you think?

-Larry

SCHEDULE 2

From: Tom Levergood
Sent: Friday, June 02, 1995 5:36 AM
To: Bill Daily
Cc: tml@OpenMarket.com; gifford@lcs.mit.edu; morris@OpenMarket.com
Subject: Re: Session ID Patent

---boundary-LibPST-iamunique-769500843_-_-
Content-type: text/plain

Bill,

Steve Morris and I read the patent app yesterday. Combined, we have a few comments to correct apparent errors. Can you give me a call Friday morning at your convenience? Thanks. (PS I have interviews between 10:00 and 11:15 so will most likely not be available during that period).

Thanks.
Tom